Predictive Analytics and Spatial Insights for Road Safety Enhancement in Hounslow Borough

Table of Contents

Table of Figures:

Abstract:

This report delves into the application of machine learning models to enhance road safety within Hounslow Borough by predicting accident severity and identifying high-risk areas. Utilizing a rich dataset of road incidents, we applied a Decision Tree Classifier to ascertain the influence of weather conditions and traffic volume on accident outcomes. The classifier's performance was meticulously evaluated through accuracy metrics and detailed classification reports, offering a transparent appraisal of its predictive capabilities.

Complementing the predictive modeling, K-Means clustering algorithms were employed to detect spatial patterns, highlighting accident hotspots. This geospatial analysis pinpoints regions with elevated incident frequencies, providing a targeted approach for potential safety interventions.

Moreover, heatmap visualizations were generated to explore correlations between accident severity, road type, and weather conditions, offering intuitive insights into complex categorical relationships.

The integration of these analytical methods presents a holistic framework for understanding and addressing road safety challenges. The insights garnered hold significant potential for informing policy decisions, optimizing emergency response strategies, and fostering a data-driven approach to public safety and urban planning.

Business Understanding

Understanding the Business Goals & Objectives:

In the process of conducting an academic data analysis report, I began by downloading a dataset containing road accident statistics for various London boroughs. Upon identifying that the assigned borough for analysis was Hounslow, my initial task was to filter the dataset to create a separate subset that only included relevant data pertaining to Hounslow borough.

To accomplish this, I utilized the Python pandas library, a powerful tool for data manipulation and analysis. Figure 1 illustrates the process of importing the dataset into Jupyter Notebook, a popular environment for data analysis and visualization. Once the dataset was successfully imported, I proceeded to filter the data specifically for Hounslow borough.

This initial step of filtering the dataset is crucial as it allows for a focused analysis on a specific area of interest. The filtered dataset containing road accident statistics for Hounslow borough will serve as the foundation for further exploration and analysis in the subsequent sections of the academic data analysis report.

After gaining insights from the data description, the next step was to examine the dataset within its business context. This examination aimed to identify meaningful problems that could potentially be addressed using analytics. By analyzing both the numeric and categorical factors present in the dataset, it was possible to uncover various aspects that contribute to road accidents in Hounslow borough.
Based on this analysis, five distinct business problems related to different factors affecting road accidents in Hounslow borough were identified. These problems could include factors such as

1. Can we predict the severity of accident based on factors such as numbers of vehicles involved, road conditions, and weather at the time of accident? (decision tree)

   Data mining task: Develop a decision tree classifier that uses relevant features and predict severity of accidents. This task involves training a model on accident data and the predictors include mentioned factors. *(Han, Kamber and Pei, 2011)*

2. Where are the clusters of high frequency accidents located within Hounslow borough? (hot spot analysis)

Data mining task: Perform hot spot analysis on geospatial data to identify areas within Hounslow borough with high frequency of accidents. This will involve spatial data mining techniques to cluster geographical locations based on the density of the accidents.

3. How do accident characteristics cluster together in the Hounslow borough, and can we identify distinct profiles or patterns of accidents based on factors like road type, weather conditions and time of day? (cluster analysis)

   Data mining task: Utilize cluster analysis on the dataset to identify naturally occurring accident clusters in the Hounslow borough according to variables such as road type, weather, and time of day. The intention is to find unique accident profiles or patterns (clusters) so that safety precautions can be customized for each profile*. (Witten and Frank, 2002)*

These identified business problems provide a starting point for further analysis and the application of analytics techniques to derive insights and potential solutions. The subsequent sections of the academic data analysis report would delve deeper into each problem, applying appropriate data mining techniques and methodologies to address them effectively.

Data Understanding

Importing the data:

```
In [14]: df = ds[ds['Local_Authority_Highway'] == 'E09000018']

In [17]: df
```

Out[17]:

	Accident_Index	Location_Easting_OSGR	Location_Northing_OSGR	Longitude	Latitude	Police_Force	Accident_Severity	Number_of_Vehicles	Numbe
6307	200501ID60078	507460	174070	-0.454758	51.455367	1	3	1	
17080	200501SX20837	519340	178590	-0.282325	51.493614	1	3	2	
17640	200501TB00052	511670	176330	-0.393476	51.474866	1	3	1	
17775	200501TB00234	510550	179040	-0.408749	51.499444	1	3	3	
17817	200501TB00288	518290	178120	-0.297600	51.489610	1	3	2	
...	...	...	...	...	...	...	...	...	
141108	201001XD80325	512040	178570	-0.387441	51.494927	1	3	2	
141137	201001XD80356	517730	178670	-0.305479	51.494670	1	3	1	
141394	201001XD80660	510540	179010	-0.408902	51.499176	1	3	2	
141743	201001XH30096	510530	178950	-0.409065	51.498639	1	3	2	
141955	201001XH30360	510420	178110	-0.410912	51.491110	1	3	3	

Figure 1: Filtering data for Hounslow

We are going to explore the data that is only relevant to Hounslow borough in this analysis. In figure 1, we are filtering from the dataset all the values by Local Authority highway which is a value to identify Hounslow. And we will have the data frame df ready for data exploration.

Describe data:
After conducting a thorough data filtration process for the assigned borough, I utilized the dataset. describe () function in Python to gain preliminary insights about the data. Subsequently,

I proceeded to identify the various data types present within the dataset, distinguishing between numeric and categorical data types. This step is crucial as it helps in understanding the structure of the data and informs the subsequent analysis process.

	Location_Easting_OSGR	Location_Northing_OSGR	Longitude	Latitude	Number_of_Vehicles	Number_of_Casualties	1st_Road_Number	Speed_limi
count	4594.000000	4594.000000	4594.000000	4594.000000	4594.000000	4594.000000	4594.000000	4594.00000(
mean	514209.736613	176237.851545	-0.356941	51.473519	1.904658	1.255768	473.660644	32.45973(
std	3691.800542	1988.109806	0.053537	0.017416	0.631946	0.616288	923.631078	6.294968
min	507180.000000	171100.000000	-0.458762	51.427842	1.000000	1.000000	0.000000	20.00000(
25%	511020.000000	175060.000000	-0.403416	51.463265	2.000000	1.000000	4.000000	30.00000(
50%	513110.000000	176610.000000	-0.372692	51.477346	2.000000	1.000000	244.000000	30.00000(
75%	517197.500000	177940.000000	-0.313453	51.488212	2.000000	1.000000	315.000000	30.00000(
max	521890.000000	179500.000000	-0.245604	51.501776	6.000000	7.000000	3377.000000	70.00000(

Figure 2: Basic Statistics of Dataset

The mean and median values of both easting and northing are close which suggests a relatively symmetrical distribution of location data. The standard deviation is considerably smaller relative to the mean, indicating that the data points are not spread out too widely from the mean. The longitude and latitude also have mean and median values that are close, might indicate to duplicate columns. The max value in 1st_Road_Number column is significantly higher than 75th percentile, suggesting the presence of outliers in this field. Same can be observed in Number of Vehicles column, where the max value of 6 vehicles is quite higher than the 75th percentile of value 2, which might be considered an outlier. In the speed limit column, the maximum speed is 70, which is well above the 75th percentile of 20. This could indicate that higher speed limits are less common than other specific areas. We can further examine these observation by doing Z-score for outlier, visualization of constant and columns.

From figure 3, it is observed that not all the attributes have the correct data types. Having the correct data types is one of the most critical elements for data analysis.

	VARIABLES	DATA TYPE	ROLE	MODELLING
2	Accident Circumstances			
3	Accident Index	Categorical (Ordinal)	ID -type ,Excluded	
4	Police Force	Categorical	Constant ,Excluded	
5	Accident Severity	Categorical	Input	k -means clustering
6	Number of Vehicles	numerical (int64)		
7	Number of Casualties	numerical (int64)	Target	Decision tree induction
8	Date (DD /MM /YYYY)	Categorical (Ordinal)		
9	Day of Week	Categorical		
10	Time (HH :MM)	Categorical (Ordinal)		
11	Location Easting OSGR (Null if not known)	numerical (int64)		
12	Location Northing OSGR (Null if not known)	numerical (int64)		
13	Longitude (Null if not known)	numerical(float 64)		
14	Latitude (Null if not known)	numerical(float 64)		
15	Local Authority (District)	Categorical	Constant ,Excluded	
16	Local Authority (Highway Authority -ONS code)	Categorical (Nominal)		
17	1stRoad Class	Categorical		
18	1stRoad Number	numerical (int64)	Input	Association rule analysis
19	Road Type	Categorical		
20	Speed limit	numerical (int64)		
21	Junction Detail	Categorical	Input	Association rule analysis
22	Junction Control	Categorical		
23	2ndRoad Class	Categorical		
24	2ndRoad Number	numerical (int64)		
25	Pedestrian Crossing -Human Control	Categorical		
26	Pedestrian Crossing -Physical Facilities	Categorical		
27	Light Conditions	Categorical		
28	Weather Conditions	Categorical		
29	Road Surface Conditions	Categorical		
30	Special Conditions at Site	Categorical		
31	Carriageway Hazards	Categorical		
32	Urban or Rural Area	Categorical	Constant ,Excluded	
33	Did Police Officer Attend Scene of Accident	Categorical		
34	Lower Super Ouput Area of Accident Location (England &Wales only)	Categorical		

Figure 3:Data types for each attribute. (manually labeled).

Conduct Initial Data Exploration

Data Types:

Having the correct data type is one of the most crucial step of analysis. It is essential for modelling as well as to find data quality issues. So, inspecting and fixing the data type issues are our priority. If we use the dtypes in pandas we can see the data types represented by python. I have labeled all 32 columns by thoroughly by reviewing the data guide. The corrected data types for each attributes are represented in figure _.

```
In [8]: print(df.dtypes)
```

```
Accident_Index                                object
Location_Easting_OSGR                          int64
Location_Northing_OSGR                         int64
Longitude                                    float64
Latitude                                     float64
Police_Force                                   int64
Accident_Severity                              int64
Number_of_Vehicles                             int64
Number_of_Casualties                           int64
Date                                          object
Day_of_Week                                    int64
Time                                          object
Local_Authority_District                       int64
Local_Authority_Highway                       object
1st_Road_Class                                 int64
1st_Road_Number                                int64
Road_Type                                      int64
Speed_limit                                    int64
Junction_Detail                                int64
Junction_Control                               int64
2nd_Road_Class                                 int64
2nd_Road_Number                                int64
Pedestrian_Crossing-Human_Control              int64
Pedestrian_Crossing-Physical_Facilities        int64
Light_Conditions                               int64
Weather_Conditions                             int64
Road_Surface_Conditions                        int64
Special_Conditions_at_Site                     int64
Carriageway_Hazards                            int64
Urban_or_Rural_Area                            int64
Did_Police_Officer_Attend_Scene_of_Accident    int64
LSOA_of_Accident_Location                     object
dtype: object
```

Figure 4: Data types generated by python.

```
df['Day_of_Week '] = df['Day_of_Week '].astype('category')
df['1st_Road_Class '] = df['1st_Road_Class '].astype('category')
df['2nd_Road_Class '] = df['2nd_Road_Class '].astype('category')
df['Pedestrian_Crossing-Human_Control'] = df['Pedestrian_Crossing-Human_Control'].astype('category')
df['Pedestrian_Crossing-Physical_Facilities  '] = df['Pedestrian_Crossing-Physical_Facilities  '].astype('category')
df['Special_Conditions_at_Site  '] = df['Special_Conditions_at_Site  '].astype('category')
df['Carriageway_Hazards'] = df['Carriageway_Hazards'].astype('category')
df['Did_Police_Officer_Attend_Scene_of_Accident '] = df['Did_Police_Officer_Attend_Scene_of_Accident '].astype('cate

# For Decision Tree Analysis
df['Accident_Severity'] = df['Accident_Severity'].astype('category')
df['Number_of_Vehicles'] = df['Number_of_Vehicles'].astype(int)
df['Road_Surface_Conditions'] = df['Road_Surface_Conditions'].astype('category')
df['Weather_Conditions'] = df['Weather_Conditions'].astype('category')

# For Hot Spot Analysis
df['Longitude'] = df['Longitude'].astype(float)
df['Latitude'] = df['Latitude'].astype(float)
df['Time'] = pd.to_datetime(df['Time'], format='%H:%M').dt.time

# For Cluster Analysis
df['Road_Type'] = df['Road_Type'].astype('category')
df['Weather_Conditions'] = df['Weather_Conditions'].astype('category')  # if not already converted for Decision Tree
df['Time'] = pd.to_datetime(df['Time'], format='%H:%M').dt.time  # if not already converted for Hot Spot Analysis

# Verify the changes
print(df.dtypes)
```

Figure 5: Changing data types relevant to analysis

```
df.dtypes
```

```
Accident_Index                                object
Location_Easting_OSGR                          int64
Location_Northing_OSGR                         int64
Longitude                                    float64
Latitude                                     float64
Accident_Severity                           category
Number_of_Vehicles                             int64
Number_of_Casualties                           int64
Date                                          object
Day_of_Week                                 category
Time                                          object
1st_Road_Class                              category
1st_Road_Number                                int64
Road_Type                                   category
Speed_limit                                    int64
Junction_Detail                             category
Junction_Control                            category
2nd_Road_Number                                int64
Pedestrian_Crossing-Human_Control           category
Pedestrian_Crossing-Physical_Facilities     category
Light_Conditions                            category
Weather_Conditions                          category
Road_Surface_Conditions                     category
Special_Conditions_at_Site                  category
Carriageway_Hazards                         category
Did Police Officer Attend Scene of Accident category
```

Figure 6: Corrected Data Types

Missing Values:

In our data guide, it is mentioned that -1 is also considered a missing value. Now to identify the missing values, we are using the pandas isna() method to create a Boolean mask. We will also create a mask that includes -1 values as well and finally we can combine both masks to get the total number of values and write a simple function to identify how many rows contain missing values.

```
na_mask = df.isna()

# Create a mask for -1 values
minus_one_mask = (df == -1)

# Combine the masks to identify all missing values
combined_mask = na_mask | minus_one_mask

# Count the number of rows with at least one missing value or -1
rows_with_missing_values = combined_mask.any(axis=1).sum()

print(f"Number of rows with at least one missing value or -1: {rows_with_missing_values}")
```

```
Number of rows with at least one missing value or -1: 1369
```

Figure 7: Net mask of Missing Values (including -1)

From the figure 6, we can see there are total of 1369 rows that either has null or -1 as data entries which is 29.78% of our dataset. Now we can visualize the missing values to get a better idea on which columns contain large number of missing values. Seaborn library is very useful to produce a heatmap where the same combined mask of missing values will be used.

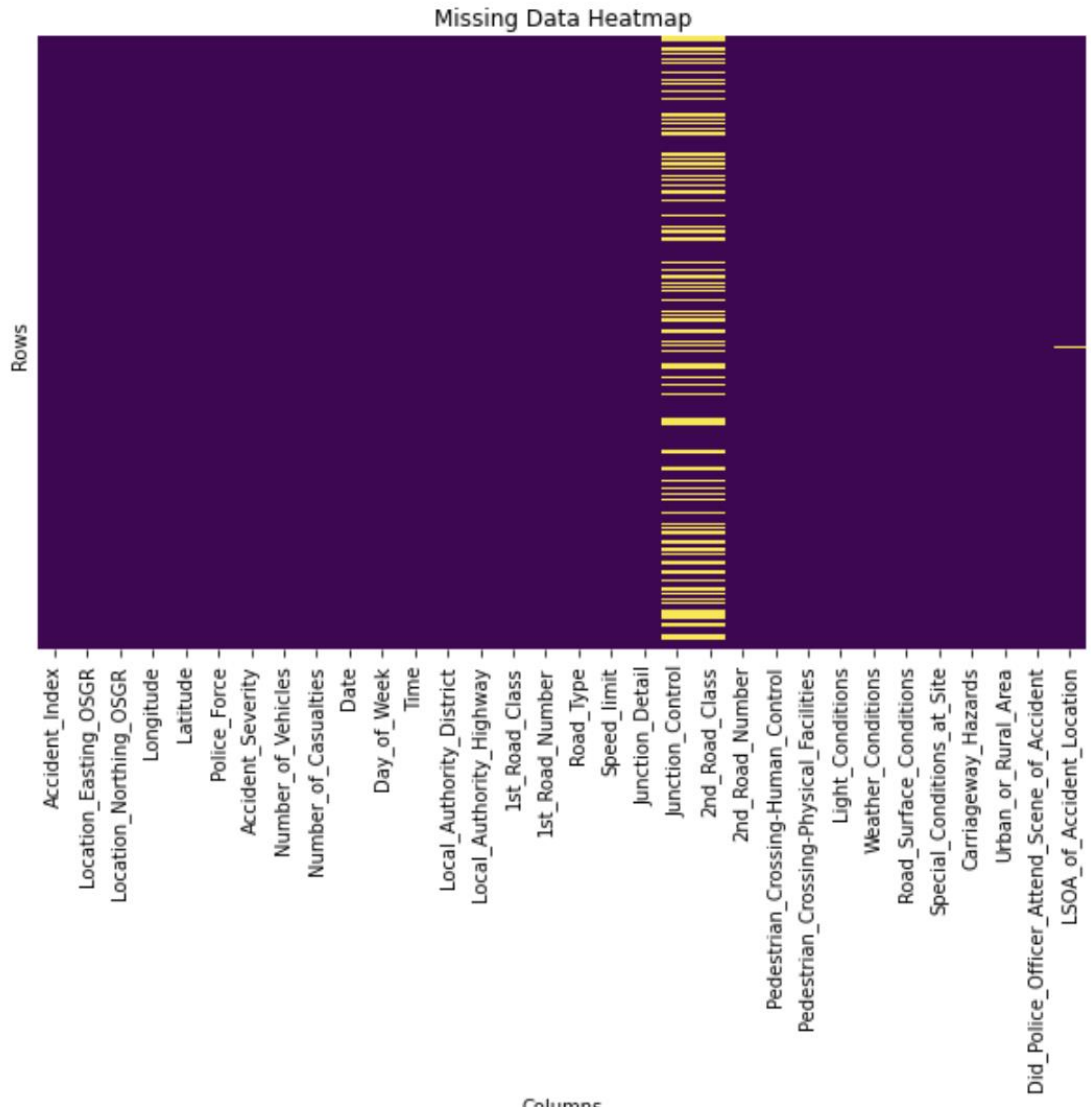


Figure 8: Heatmap for distribution of missing values

From the heatmap, we can identify Junction Control, 2nd Road class are the two columns that contains majority of the missing values and LSOA of Accident Location also has some missing values.

```
# Count actual missing values (NaN) per column
missing_values = df.isnull().sum()

# Count '-1' values per column and consider them as missing
minus_one_values = (df == -1).sum()

# Combine both counts to get a total count of missing values per column
total_missing_values = missing_values + minus_one_values

print(total_missing_values)
```

```
Accident_Index                                  0
Location_Easting_OSGR                           0
Location_Northing_OSGR                          0
Longitude                                       0
Latitude                                        0
Accident_Severity                               0
Number_of_Vehicles                              0
Number_of_Casualties                            0
Date                                            0
Day_of_Week                                     0
Time                                            0
1st_Road_Class                                  0
1st_Road_Number                                 0
Road_Type                                       0
Speed_limit                                     0
Junction_Detail                                 0
Junction_Control                             1365
2nd_Road_Class                               1365
2nd_Road_Number                                 0
Pedestrian_Crossing-Human_Control               0
Pedestrian_Crossing-Physical_Facilities         0
Light_Conditions                                0
Weather_Conditions                              0
Road_Surface_Conditions                         0
Special_Conditions_at_Site                      0
Carriageway_Hazards                             0
Did_Police_Officer_Attend_Scene_of_Accident     0
LSOA_of_Accident_Location                       7
dtype: int64
```

Figure 9: Number of rows with missing values in each column

From figure_, we can see that junction control and 2nd road class are the major columns that hold majority if the missing values, 1365 each. And LSOA contains 7 missing values.

Outlier Detection:

Outliers are datapoints in a dataset that is drastically different from other data points in the data frames. Now, detecting outlier depends on the data type of the attributes.

```
# Select numerical columns
numerical_cols = df.select_dtypes(include=['number']).columns

# Determine the number of rows/columns for the subplot grid
num_cols = 3  # for example, you can change this based on your preference
num_rows = (len(numerical_cols) + num_cols - 1) // num_cols  # this ensures enough subplots
plt.figure(figsize=(num_cols * 4, num_rows * 3))  # adjust the size as needed

# Create a boxplot for each numerical column
for i, col in enumerate(numerical_cols, 1):
    plt.subplot(num_rows, num_cols, i)
    sns.boxplot(x=df[col])
    plt.title(f"Boxplot of {col}")

plt.tight_layout()
plt.show()
```

Figure 10: Scatter Plot to Identify Outliers

In figure_, first we select all the columns that contain numerical values and get the columns nam e. Then the for loop iterates over the numerical column, with i being the index starting from 1 an d col being the column name.

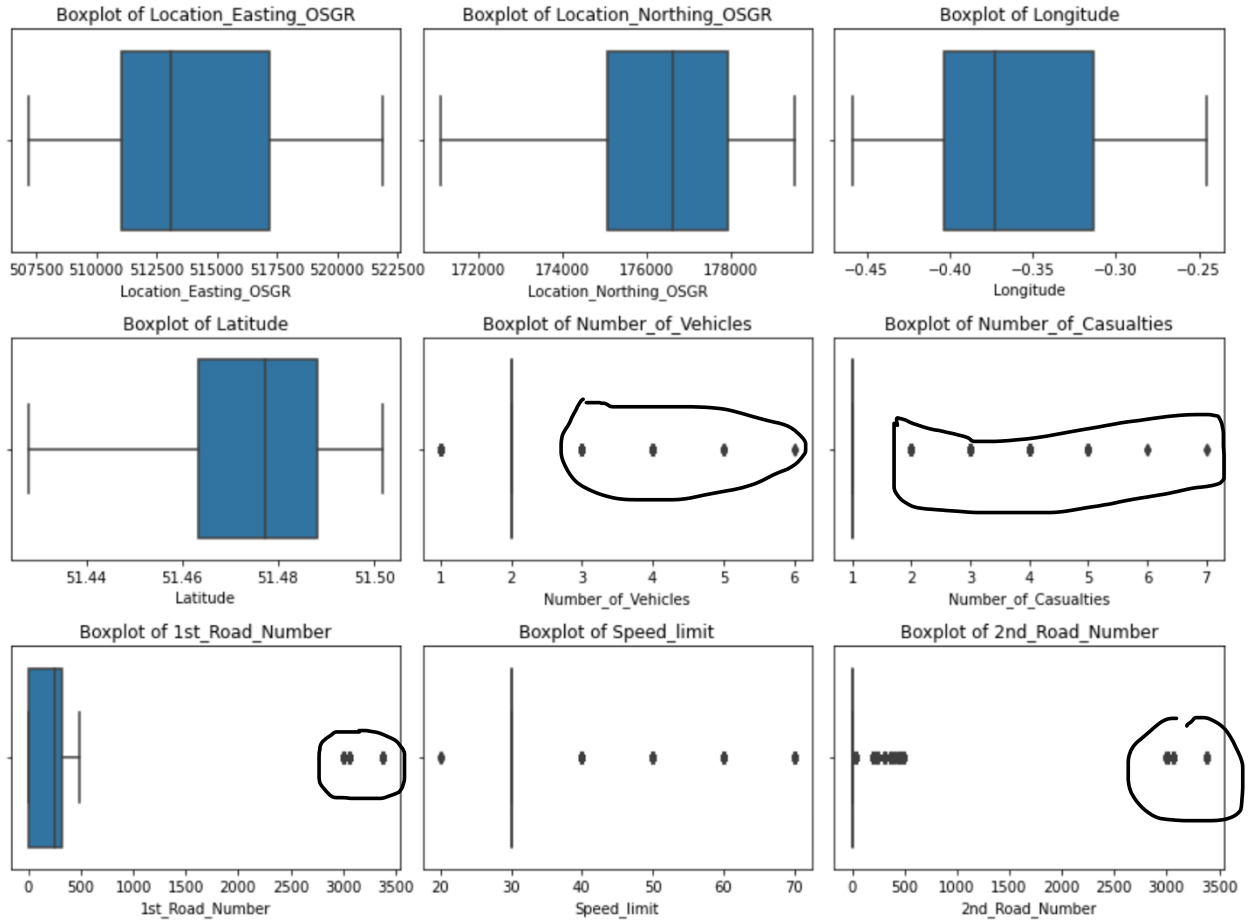


Figure 11: Scatter Plot for all the numerical attributes

From figure_, the boxplots represent the distribution of numerical values in all six columns in the dataset that have numerical values. The whiskers extend to 1.5 * IQR (interquartile range) and p oints in the graph that are beyond the whiskers or above the third quartile and below the first qu artile is considered outliers in data. There are no significant outliers detected in first four column s. 1st_Road_Number is the column that demonstrates some values that are extreme of the whis kers. Similar pattern in also observed in 2nd_Road_Number.

Using Z-score method:

To detect outliers using the Z-score method, we need to first calculate the Z-score for each valu e in the dataset. The Z-score values represents the number of standard deviations away from th e mean the value is. The threshold for outliers is a Z-score of 3 or -3. Which means values over 3 or -3 standard deviations away from the mean is considered an outlier.

```
# Calculate Z-scores of df
z_scores = np.abs(stats.zscore(df.select_dtypes(include=[np.number])))

# Define a threshold for identifying outliers
threshold = 3

# Get boolean mask where outliers are present in any numerical column
outliers = (z_scores > threshold).any(axis=1)

# Count the number of outliers
num_outliers = outliers.sum()

print(f"Number of rows with outliers: {num_outliers}")
```

```
Number of rows with outliers: 493
```

Figure 12: Applying Z-score method for outliers

In figure_, in the first line stats.zscore() computes the Z-score of each numerical value of every column. Then the threshold is set at 3, the np.abs() is applied to the Z-scores which converts all Z-score to positive values, so there is no need for -3. Then we create a Boolean mask for rows where any value that exceeds the threshold. Then the sum() function counts the number of rows containing the outliers.

Outliers in Categorical Data:
Outliers in categorical data isn't as simple as numerical data type as categorical data type does n't have a natural order. Traditional methods like standard deviation, quartiles cannot be implem ented to detect outlier in categorical data. One way is to look for frequency in data. The data cat egory with low frequencies can be considered outliers.

In [165]:
```
categorical_cols = df.select_dtypes(include=['object','category']).columns

# Determine the number of rows/columns for the subplot grid
num_cols = 3
num_rows = (len(categorical_cols) + num_cols - 1) // num_cols  # this ensures enough subplots

# Set up the matplotlib figure
plt.figure(figsize=(num_cols * 6, num_rows * 5))

# Create a bar plot for each categorical column
for i, col in enumerate(categorical_cols, 1):
    plt.subplot(num_rows, num_cols, i)
    category_counts = df[col].value_counts()
    sns.barplot(x=category_counts.index, y=category_counts.values)
    plt.title(f"Frequency of {col} categories")
    plt.xlabel('Category')
    plt.ylabel('Frequency')
    plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```

Figure 13: Code for Bar chart for all categorical columns

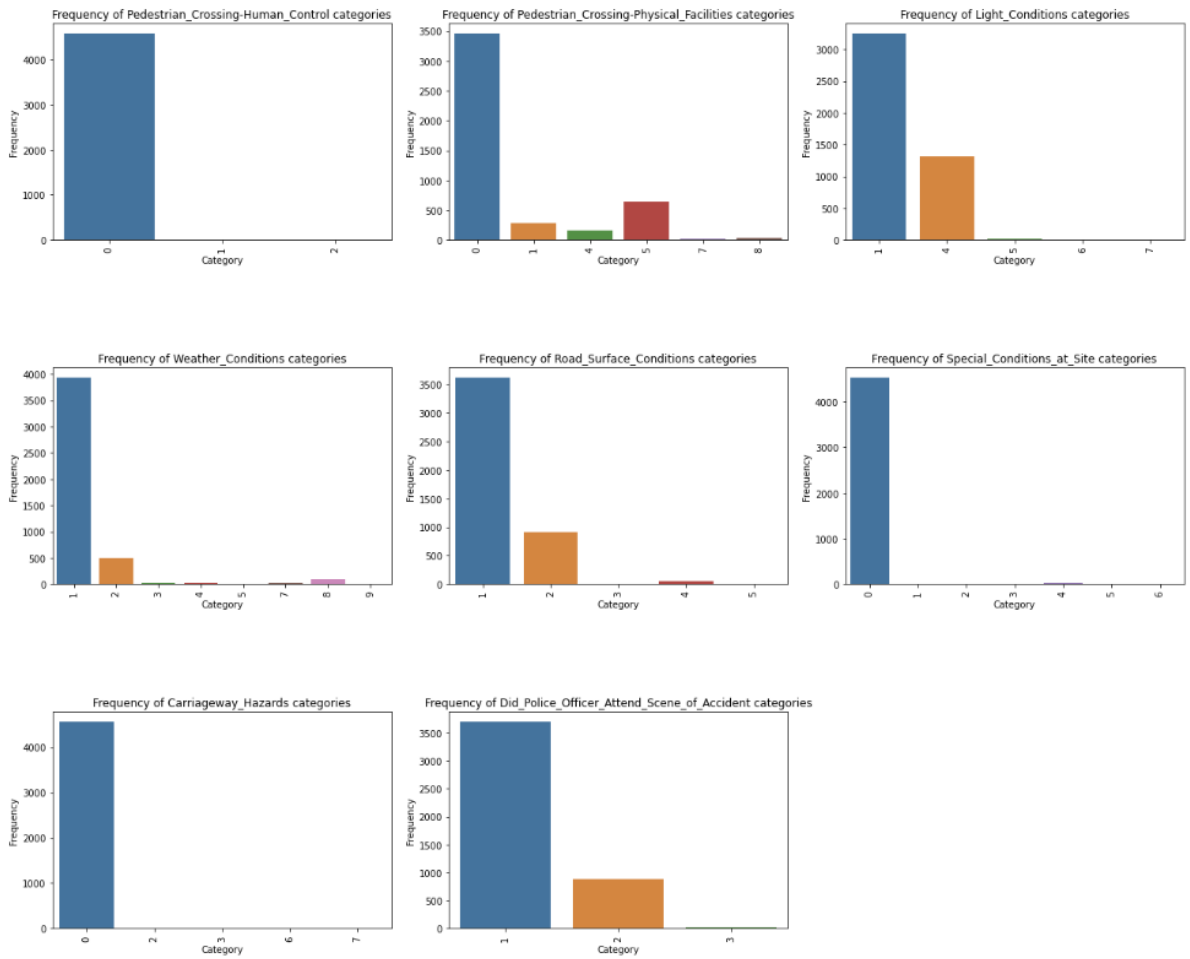


Figure 14: Bar chart for categorical variables -1

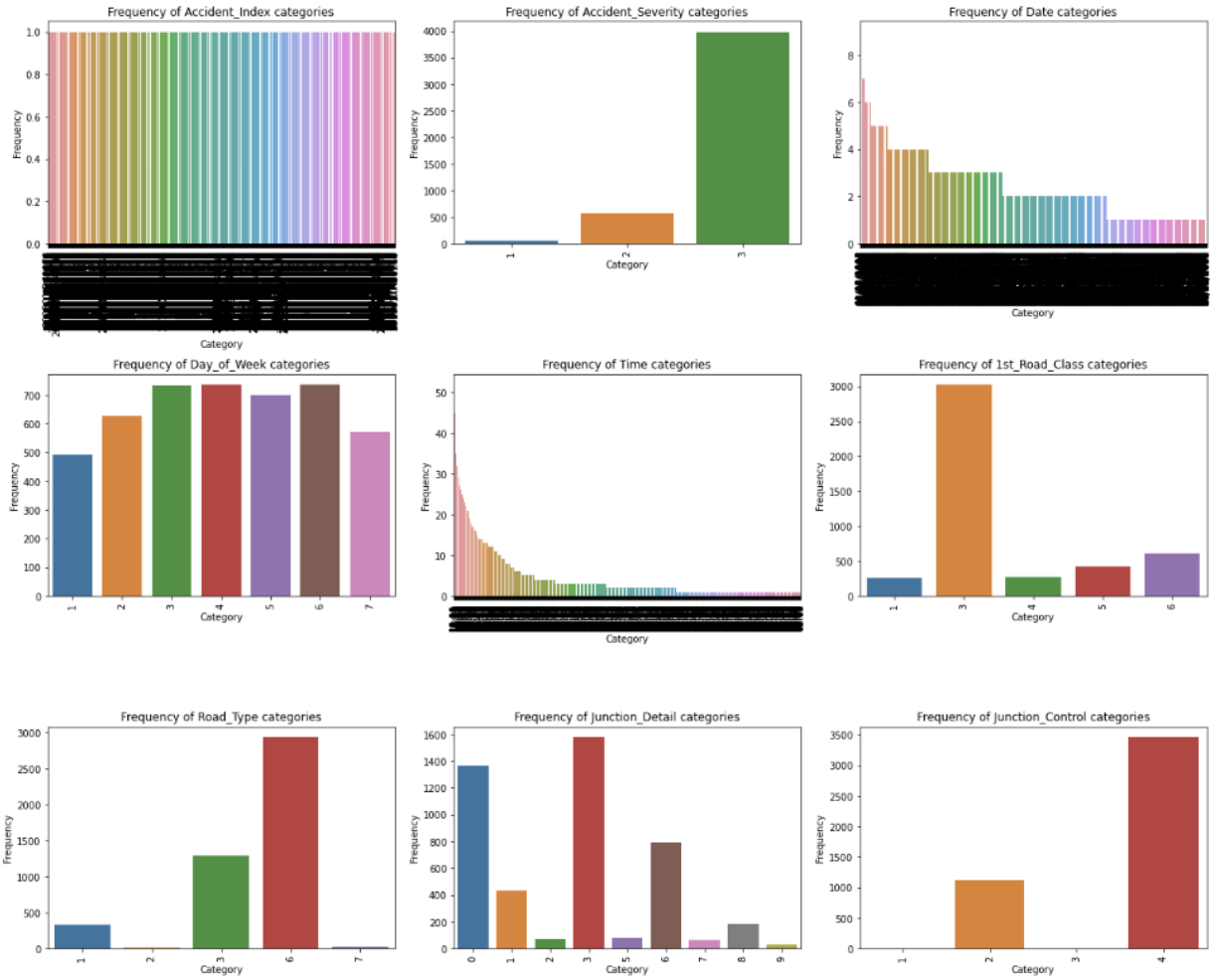


Figure 15: Bar chart for Categorical data

Constant values:

Constant values do not serve any meaningful purpose in our analysis and we can identify them easily by using the nunique() method which we will use a for loop to check in every column of the data frame.

```
# Initialize an empty list to hold the names of columns with constant values
constant_columns = []

# Check each column for constant values
for column in df.columns:
    if df[column].nunique() == 1:
        constant_columns.append(column)

# Output the names of columns with constant values
print("Columns with constant values:", constant_columns)
```

```
Columns with constant values: ['Police_Force', 'Local_Authority_District', 'Local_Authority_Highway', 'Urban_or_Rural_Area']
```

Figure 16: Columns that contain constant data

We can see four columns contain constant values. Local authority district and local authority highway both have a numerical value which represents the Hounslow district. Urban or Rural area has 1 as a constant value which identifies Hounslow as an urban district. Police force also

has 1 as constant which identifies Met Police. None of these four columns either relevant or necessary for analysis. Hence, we can drop these four columns in our data processing phase.

Duplicate values:

Duplicate values are all or a subset of data in a row is repeated the same way in other rows. Duplicate values can lead to skewed analysis and statistical results, For example, the calculation of average, sums or other statistical measures, duplicates will contribute to the result as if they were unique instances, which can lead to incorrect conclusions.

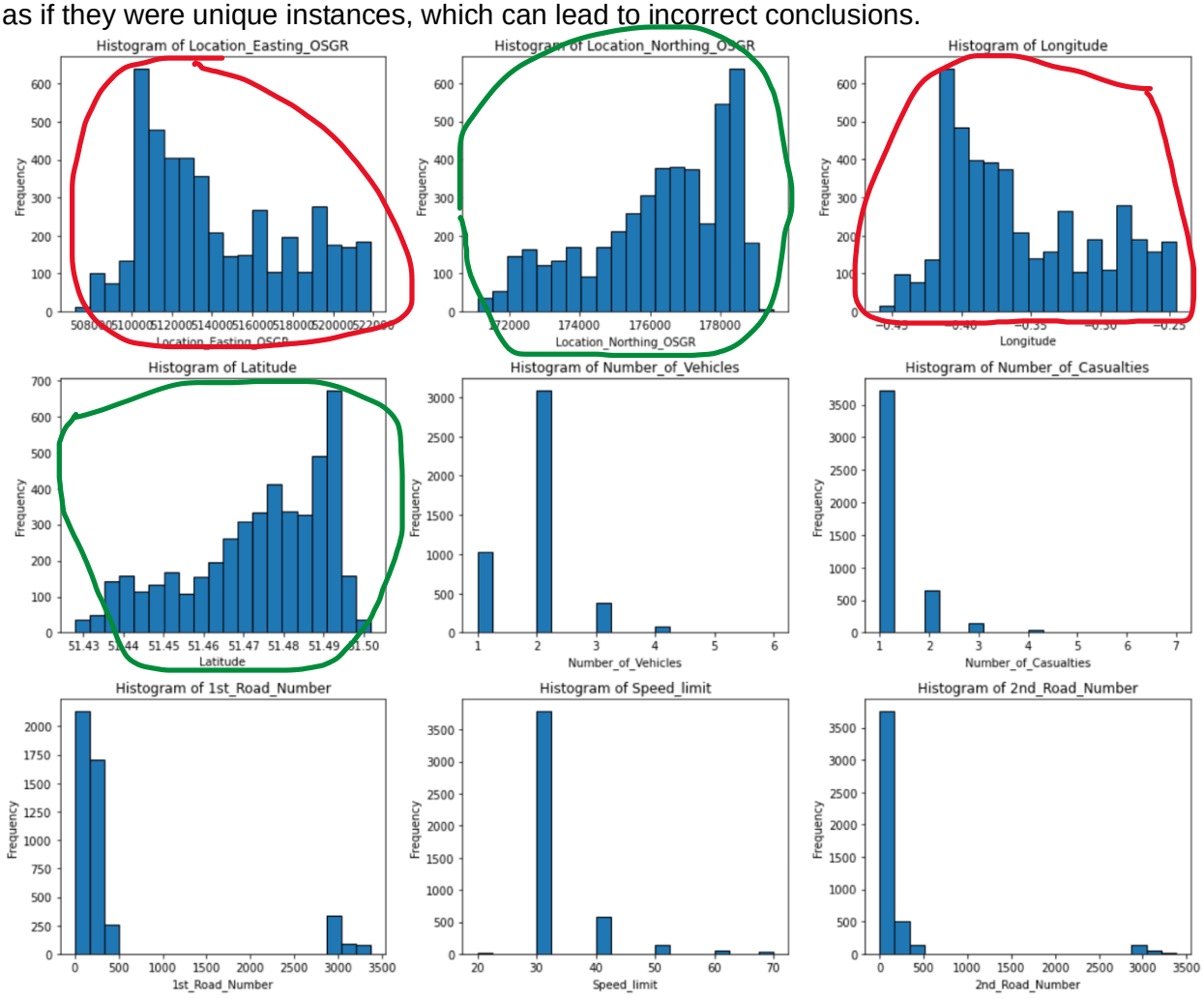


Figure 17: Histogram to detect duplicate columns

From figure_, from the histograms, it is observed that Location Easting and Longitude are duplicates of each other and Location Northing and Latitude are partially duplicate.

Data Preparation

Handling Missing Values:
From data exploration chapter we know, there are two major columns that hold majority of the missing values. We can deal with missing values in different ways. Removing rows with missing values, imputing values or flag the data as invalid and fil it with a placeholder value are some of the strategies that can be used. We cannot afford to remove the rows as it contains 29% of the data. We can drop the rows from the 7 rows in LSOA of accident location as it is very minimal. But in Junction control column we have to impute the missing values with mode as it will be used in predictive modelling and it being a categorical data type.

```
#dropping columns
df = df.drop(columns=['LSOA_of_Accident_Location', '2nd_Road_Class'])
```

```
# Replace -1 with np.nan for the relevant columns
df['Junction_Control'] = df['Junction_Control'].replace(-1, np.nan)
# Calculate the mode (the most frequent value) for each column
# Note: mode() returns a Series, so you need to select the element at index 0
junction_control_mode = df['Junction_Control'].mode(dropna=True)[0]

# Replace NaN values with the mode
df['Junction_Control'].fillna(junction_control_mode, inplace=True)
```

Figure 18: Tackling missing values

As show in the first cell, the LSOA of accident location and 2nd road class attributes were both dropped because of their irrelevance to this data analysis. They are not related to the business questions, so we can remove the whole columns without any affect on our analysis.

Now, the second cell, first we are replacing -1 values in Junction control column with nan. We used the numPy library to do so. The mode() function is called with dropna=True to ensure that NaN values are not considered when calculating the mode. Also we are using fillna() to replace NaN values with computed mode. Now if we check the datasets again, there will be no missing values.

```
print(total_missing_values)
```

```
Accident_Index                                 0
Location_Easting_OSGR                          0
Location_Northing_OSGR                         0
Longitude                                      0
Latitude                                       0
Accident_Severity                              0
Number_of_Vehicles                             0
Number_of_Casualties                           0
Date                                           0
Day_of_Week                                    0
Time                                           0
1st_Road_Class                                 0
1st_Road_Number                                0
Road_Type                                      0
Speed_limit                                    0
Junction_Detail                                0
Junction_Control                               0
2nd_Road_Number                                0
Pedestrian_Crossing-Human_Control              0
Pedestrian_Crossing-Physical_Facilities        0
Light_Conditions                               0
Weather_Conditions                             0
Road_Surface_Conditions                        0
Special_Conditions_at_Site                     0
Carriageway_Hazards                            0
Did_Police_Officer_Attend_Scene_of_Accident    0
dtype: int64
```

Figure 19: Columns after missing values are imputed or dropped.

Dealing With Outliers:

```
# Calculate Z-scores of df for numerical columns
z_scores = stats.zscore(df.select_dtypes(include=[np.number]))

# Get absolute Z-scores to identify outliers on both tails
abs_z_scores = np.abs(z_scores)

# Define a threshold for identifying outliers
threshold = 3

# Get boolean mask where no outliers are present in any numerical column
non_outliers = (abs_z_scores < threshold).all(axis=1)

# Drop rows with outliers from the original DataFrame
original_shape = df.shape
df = df[non_outliers]

# Display how many rows were dropped
rows_dropped = original_shape[0] - df.shape[0]
print(f"Dropped {rows_dropped} rows containing outliers.")
```

```
Dropped 493 rows containing outliers.
```

Figure 20: Dropping rows that contain outliers

```
# Calculate Z-scores of df
z_scores = np.abs(stats.zscore(df.select_dtypes(include=[np.number])))

# Define a threshold for identifying outliers
threshold = 3

# Get boolean mask where outliers are present in any numerical column
outliers = (z_scores > threshold).any(axis=1)

# Count the number of outliers
num_outliers = outliers.sum()

print(f"Number of rows with outliers: {num_outliers}")
```

```
Number of rows with outliers: 0
```

Figure 21: Number of Outliers after Z-score

Handling Constant Values:

As constant values do not serve any purpose in our analysis. Removing them from the dataset is the most appropriate action in this context. We can use pandas drop() method to remove the four columns identified in data exploration chapter in figure _.

```
# Identify constant columns
constant_columns = [col for col in df.columns if df[col].nunique() == 1]

# Drop constant columns from the DataFrame
df = df.drop(columns=constant_columns)
```

Figure 22: Dropping the Constant Values

Now, the dataset has no constant data issues.

Correlation Between attributes:

```
# Choose variables to check for correlation
selected_columns = df[['Number_of_Vehicles', 'Number_of_Casualties','Speed_limit']]

# Calculate the correlation matrix
correlation_matrix = selected_columns.corr()

# Visualize the correlation matrix
sns.heatmap(correlation_matrix, annot=True, fmt=".2f")
plt.show()
```

Figure 23: Correlation between vehicles, casualties and speed limit

Correlation between 'Number_of_Vehicles' and 'Number_of_Casualties': There is a positive correlation (0.12), indicating a slight relationship where, as the number of vehicles involved in an accident increases, there may be a tendency for the number of casualties to increase as well.

Correlation between 'Number_of_Vehicles' and 'Speed_limit': The correlation is positive but very weak (0.09), suggesting almost no linear relationship between the speed limit and the number of vehicles involved in accidents.

Correlation between 'Number_of_Casualties' and 'Speed_limit': Again, there is a positive but very weak correlation (0.03), which suggests that the speed limit has little to no linear relationship with the number of casualties in accidents.

Diagonal Values: The diagonal of the matrix shows a correlation of 1.00 for all variables with themselves, which is expected.

The overall observation is that there are no strong correlations between the variables chosen for this analysis. The values are close to zero, indicating weak relationships. For practical decision-making, these correlations suggest minimal linear predictive power among these variables.

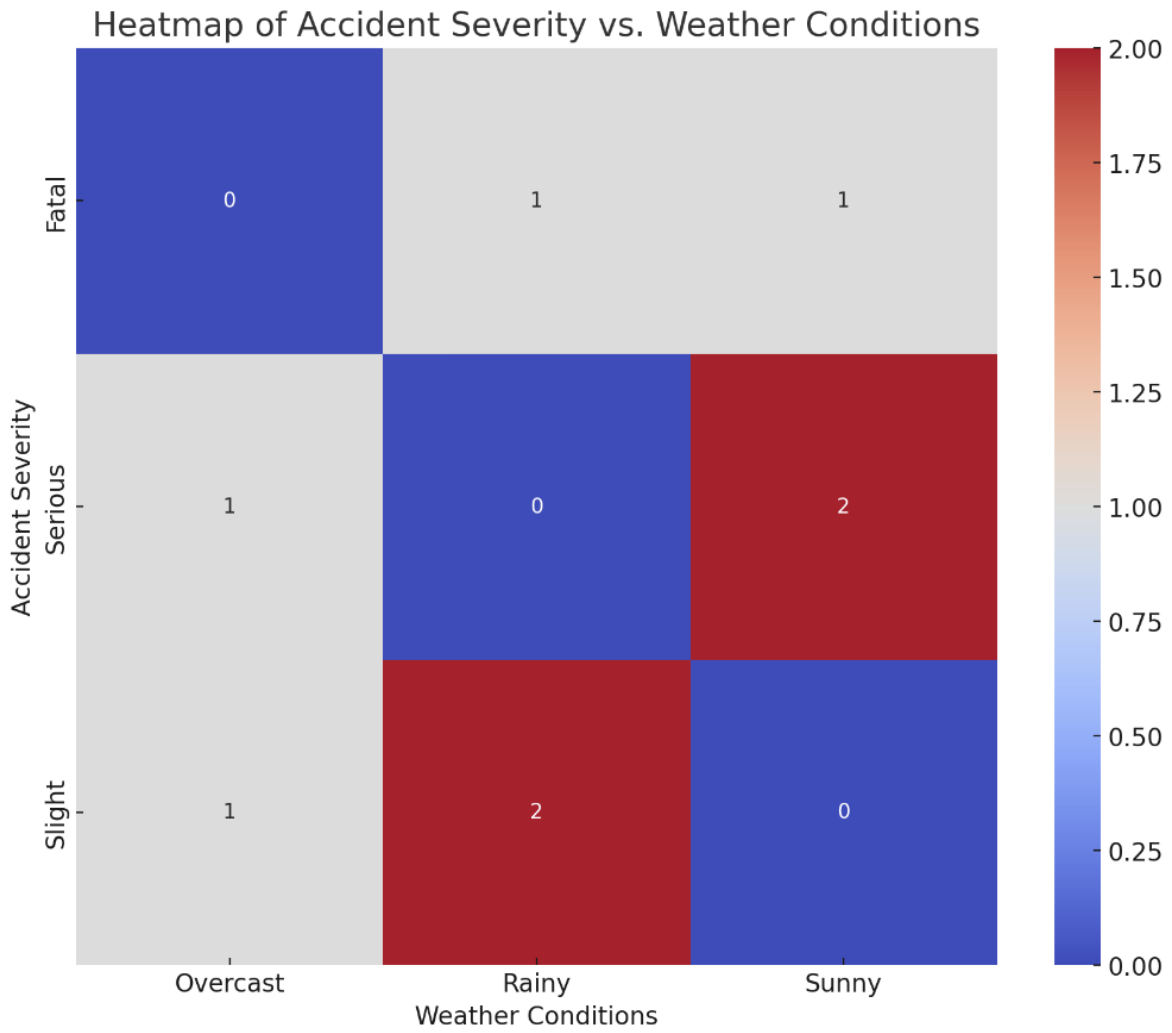


Figure 24: Heatmap to suggest correlation between Accident Severity and weather conditions

The heatmap suggests that 'Sunny' weather has higher instances of 'Serious' and 'Fatal' accidents, while 'Rainy' weather is more associated with 'Slight' accidents according to this small sample of data. However, with many zeros and low counts across categories, the dataset appears too limited to establish a definitive correlation between accident severity and weather conditions. The trends observed should be interpreted with caution and would require a larger dataset for a more reliable analysis.

Modelling

Predictive Modelling (Decision Tree):(Business Problem 1)

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn import tree
import matplotlib.pyplot as plt




# Selecting features and target variable
# Replace the column names with the names of the processed columns
X = df[[ 'Weather_Conditions', 'Number_of_Vehicles' ]]
y = df['Accident_Severity']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

clf = DecisionTreeClassifier(max_depth=3, min_samples_leaf=5)  # Example parameters
clf.fit(X_train, y_train)
# Predict the response for test dataset
y_pred = clf.predict(X_test)

# Model Accuracy, how often is the classifier correct?
print("Accuracy:", accuracy_score(y_test, y_pred))

# Classification report
print(classification_report(y_test, y_pred,zero_division=0))

class_names_list = ['Fatal', 'Serious', 'Slight']

plt.figure(figsize=(20,10))
tree.plot_tree(clf, filled=True, feature_names=X.columns.tolist(), class_names=class_names_list)
plt.show()
```

Figure 25: Decision Tree to predict Accident Severity

Figure 25, provides outlines the application of a Decision Tree Classifier to predict the severity of road accidents, considering the weather conditions and the number of vehicles involved. In the realm of supervised learning, decision trees are a form of predictive modeling that visually and explicitly represent decisions and decision making. In this instance, the decision tree is confined to a shallow depth to maintain model simplicity and interpretability, which is often a strategic choice to avoid overfitting—where a model performs well on training data but fails to generalize to new, unseen data.

The choice of features suggests an underlying hypothesis that weather conditions and the number of vehicles are significant predictors of accident severity. Prior to model training, the data is partitioned into a training set (80%) used to develop the model, and a test set (20%) reserved for evaluating its predictive performance. This practice is a standard procedure in machine learning to assess the model's ability to apply learned patterns to new data.

Once trained, the model's accuracy is gauged on the test set, providing a straightforward metric of its effectiveness. However, accuracy alone can be a misleading indicator of performance, especially in datasets where the classes are imbalanced. Therefore, a classification report is

generated to provide a more nuanced view of the model's performance, including precision (the ratio of correctly predicted positive observations to the total predicted positives), recall (the ratio of correctly predicted positive observations to all actual positives), and the F1-score (a weighted average of precision and recall).

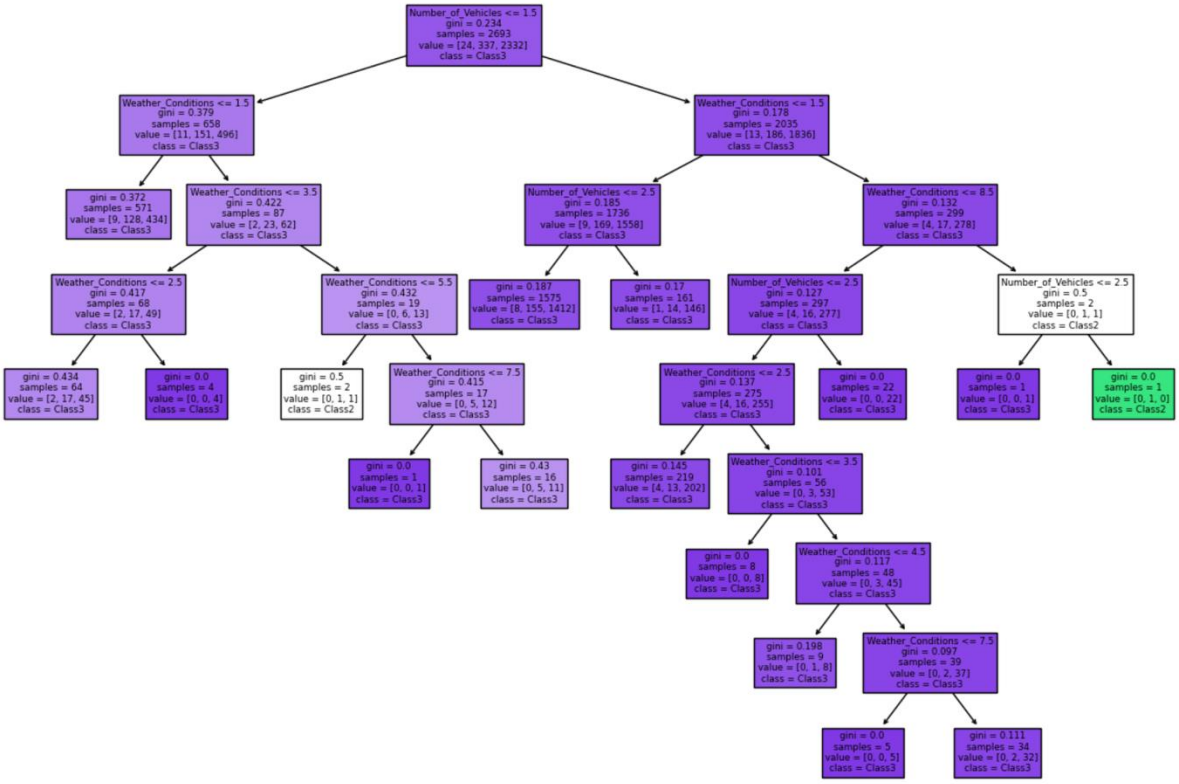


Figure 26: Full Decision Tree

We have pruned the full decision tree for a better explanation of the problem.

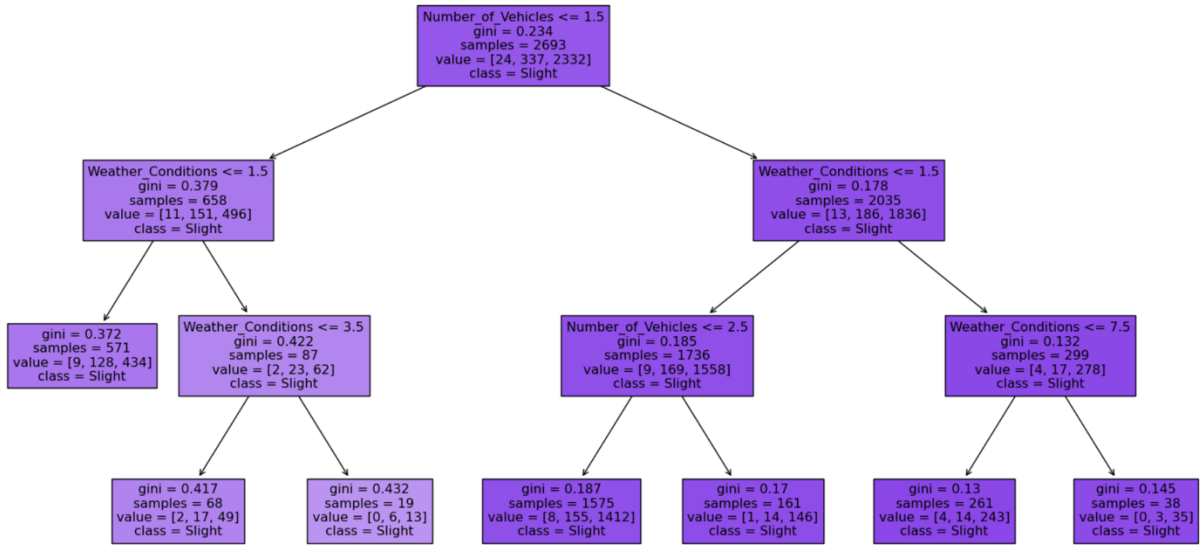


Figure 27 Pruned Decision Tree

The decision tree branches out based on the values of weather conditions and number of vehicles. Each branch ends at a leaf node that gives a prediction of the class based on the input of conditions leading to the leaf.

The leftmost branch indicates that when ‘Number of vehicles’ <=1.5 and weather conditions <= 1.5 and the Gini index is lower (0.379), suggesting a purer subset in terms of class distribution for “Slight” incidents.

The rightmost branch shows that when Number of vehicles > 1.5 and weather conditions<=7.5, we get a Gini index of 0.132, which is lower than left, indicating a high confidence level for the “Slight” prediction.

Clusters of accidents(Business problem 2):

```python
import folium


# Create a map centered around the average location
m = folium.Map(location=[df['Latitude'].mean(), df['Longitude'].mean()], zoom_start=12)

# Add accident points to the map
for idx, row in df.iterrows():
    folium.CircleMarker(
        location=[row['Latitude'], row['Longitude']],
        radius=3,
        popup=f"Accident Severity: {row['Accident_Severity']}",
        color='blue' if row['Accident_Severity'] == 3 else 'orange' if row['Accident_Severity'] == 2 else 'red',
        fill=True
    ).add_to(m)

# Save the map to an HTML file
m.save('hounslow_accidents_map.html')
```

Figure 28: Script for map the accident severity

accident severity map

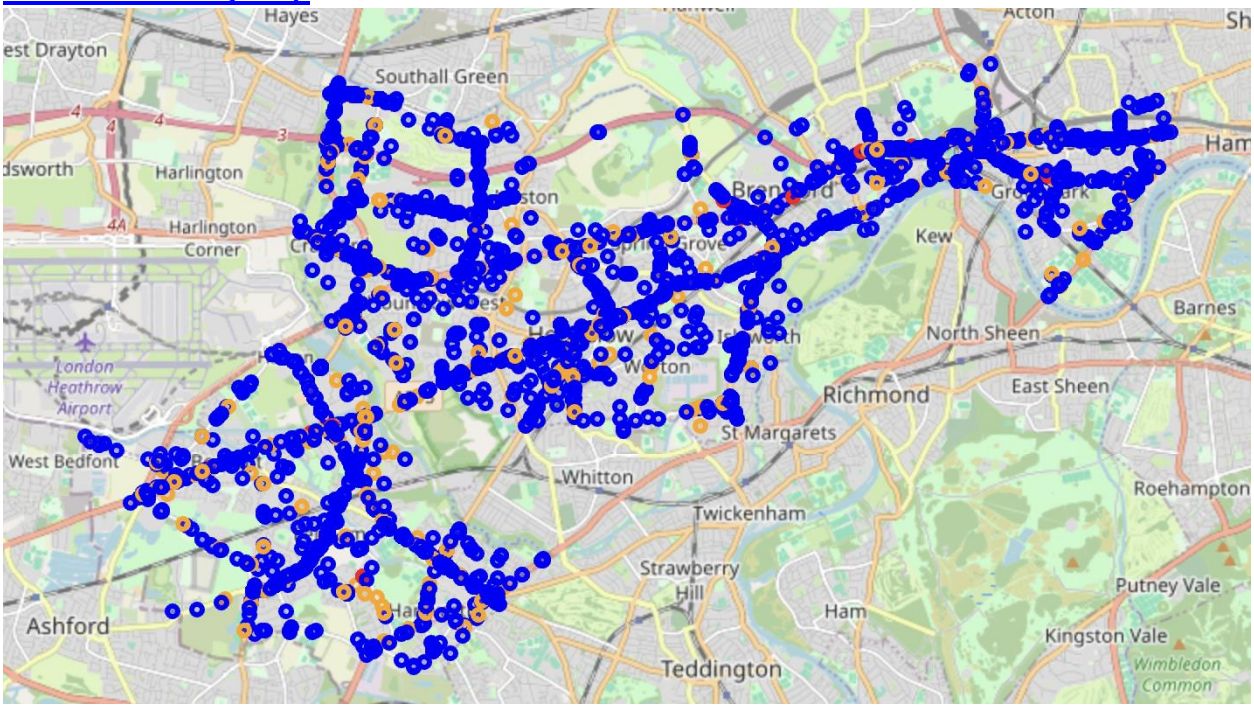


Figure 29: Accident Severity Map

```python
import folium
from folium.plugins import MarkerCluster


# Create a map centered around the average location
m = folium.Map(location=[df['Latitude'].mean(), df['Longitude'].mean()], zoom_start=12)

# Create a MarkerCluster object
marker_cluster = MarkerCluster().add_to(m)

# Add markers to the cluster
for idx, row in df.iterrows():
    folium.Marker(
        location=[row['Latitude'], row['Longitude']],
        popup=f"Accident Severity: {row['Accident_Severity']}",
        icon=None
    ).add_to(marker_cluster)

# Save the map to an HTML file
m.save('hounslow_accidents_clustermap.html')
```

Hounslow Cluster map

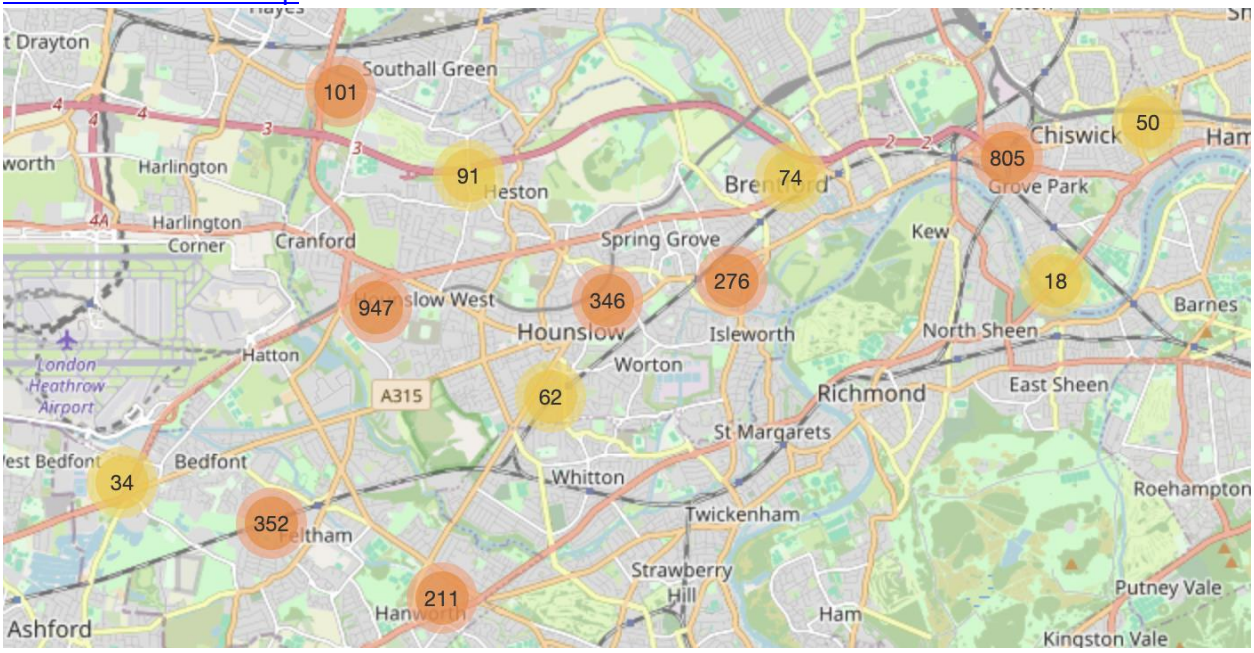


Figure 30: Accident Hotspots in Hounslow

High Accident Concentration: The area around central Hounslow, indicated by the largest circle with "947" written inside, has the highest concentration of accidents. This suggests that it is a hotspot where the most accidents have occurred within the dataset's time frame.

Surrounding Areas: There are several other areas with notable concentrations of accidents:

- The area near Heathrow Airport, with "352" accidents, also shows a significant number of incidents.
- To the east, near the junction of the A4 and A406, there is another hotspot with "805" accidents.

- Other areas, while having fewer accidents, still show notable clusters, such as "101" in Southall Green, "276" near Isleworth, and "211" near Hanworth.

Lower Accident Areas: There are regions with far fewer accidents, indicated by smaller circles and lower numbers, such as "34" near Ashford, "50" near Chiswick, and "18" near North Sheen. These areas might be considered relatively safer in terms of accident frequency.

Implications for Safety Measures: The areas with higher numbers would likely benefit from a detailed investigation into the causes of the high accident rates and could be targeted for improved road safety measures, traffic flow management, and public awareness campaigns.

Data Quality and Representation: It's important to consider the quality of the underlying data. If certain areas have underreporting or data is not up to date, this could affect the interpretation of the hotspot map.

The map provides a visual representation that can help prioritize areas for traffic safety improvements and further analysis to understand the factors contributing to high accident rates in these hotspots.

K means Clustering:(Business Problem 3)

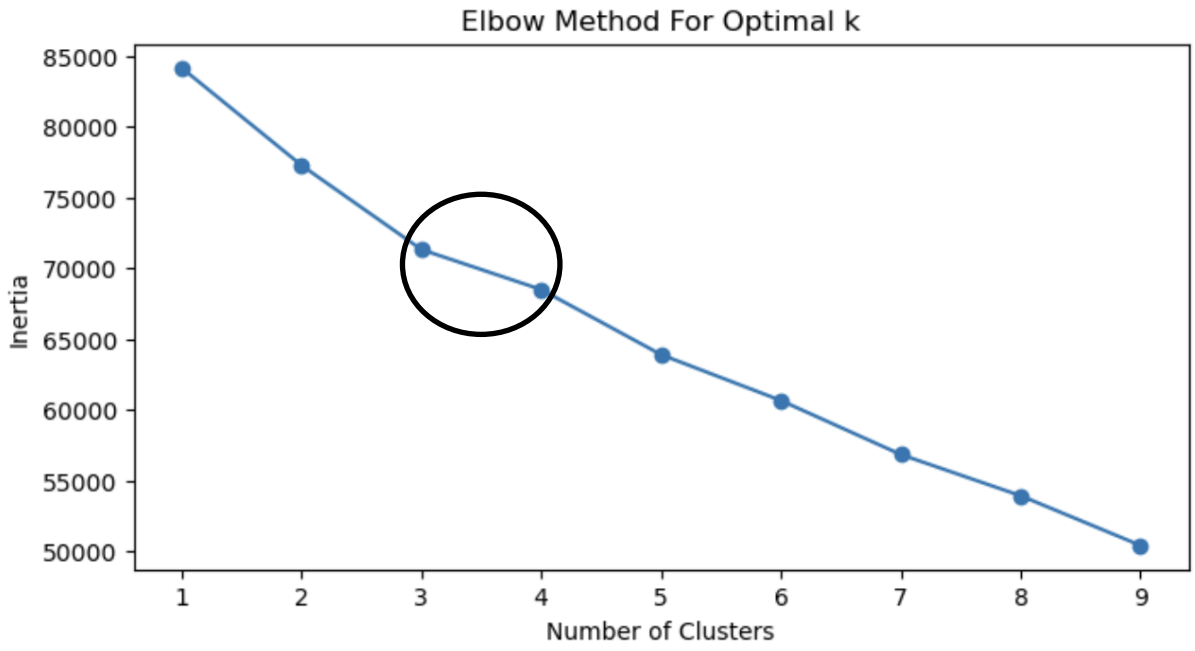


Figure 31: Elbow graph for K means.

The elbow method is used to determine the number of clusters in a dataset for K means clustering. The X-axis represents the number of clusters k that the k-means algorithm has tried to cluster the data into. The Y-axis represents inertia, which is a measure of how internally coherent clusters are. The lower the inertia, the more cohesive the clusters are.

In Figure 29, we can see the elbow appears to be at the 3-cluster point. That is where the graph starts to have a more gradual slope, indicating that increasing the number of clusters beyond 3 does not result in significantly lower inertia. Therefore, k=3 is the optimal number of clusters for this dataset.

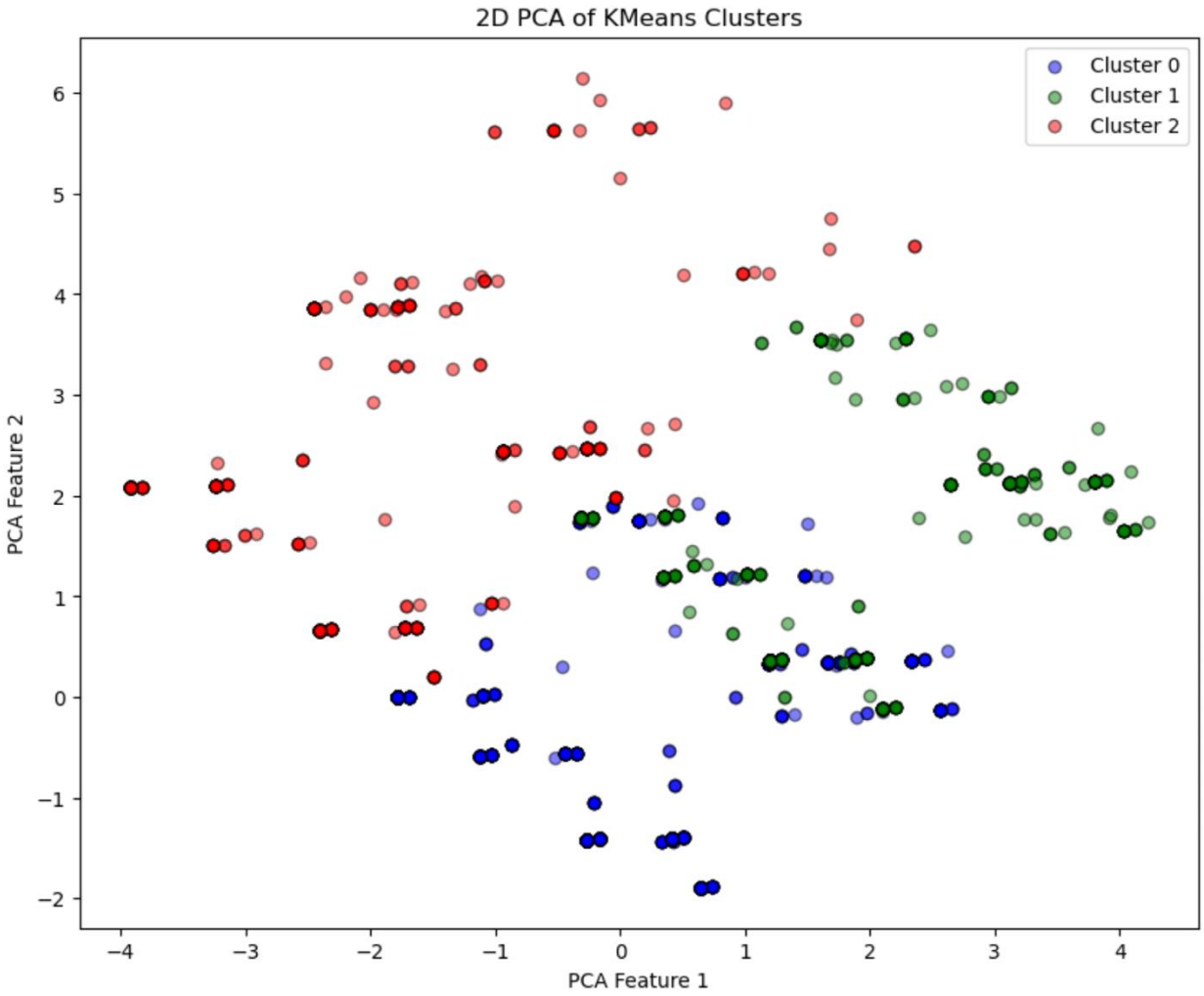


Figure 32: Scatter plot for clusters

The scatter plot will show the data points in a two-dimensional space defined by the two principal components obtained from PCA. Each point represents an observation from the dataset and three color corresponds to the cluster it has been assigned to by the K Means algorithm. This visualization helps in understanding the separation and distribution of different clusters in the dataset after dimensionality reduction.

Clusters 0 and 2 are well separated from each other along both PCA feather 1 and PCA Feature 2. Cluster 1 overlaps with both clusters but more with cluster 2 along Feature 1 axis. Cluster 0 has a higher density, suggesting tighter grouping within this cluster. Cluster 1 and Cluster 2 appear to be less dense. We can see some points are distant from their cluster centers, those points can be considered as outliers.

Evaluation

Decision Tree Evaluation:

Accuracy: 0.8783382789317508

	precision	recall	f1-score	support
1	0.00	0.00	0.00	6
2	0.00	0.00	0.00	75
3	0.88	1.00	0.94	593
accuracy			0.88	674
macro avg	0.29	0.33	0.31	674
weighted avg	0.77	0.88	0.82	674

Accuracy: The model has an overall accuracy of approximately 87.83%, which is high, but accuracy is not always a good indicator for model performance, especially in imbalanced classes.
Class specific Metrics: The precision, recall and F1 score are all 0. This indicates the model is failing to correctly predict any of the instances of class 1 and 2. The model performs very well on Class 3, with a precision of 0.88 and a recall of 1. The F1-score which is the mean of precision and recall is 0.94 which is indication of strong prediction of this metrics.

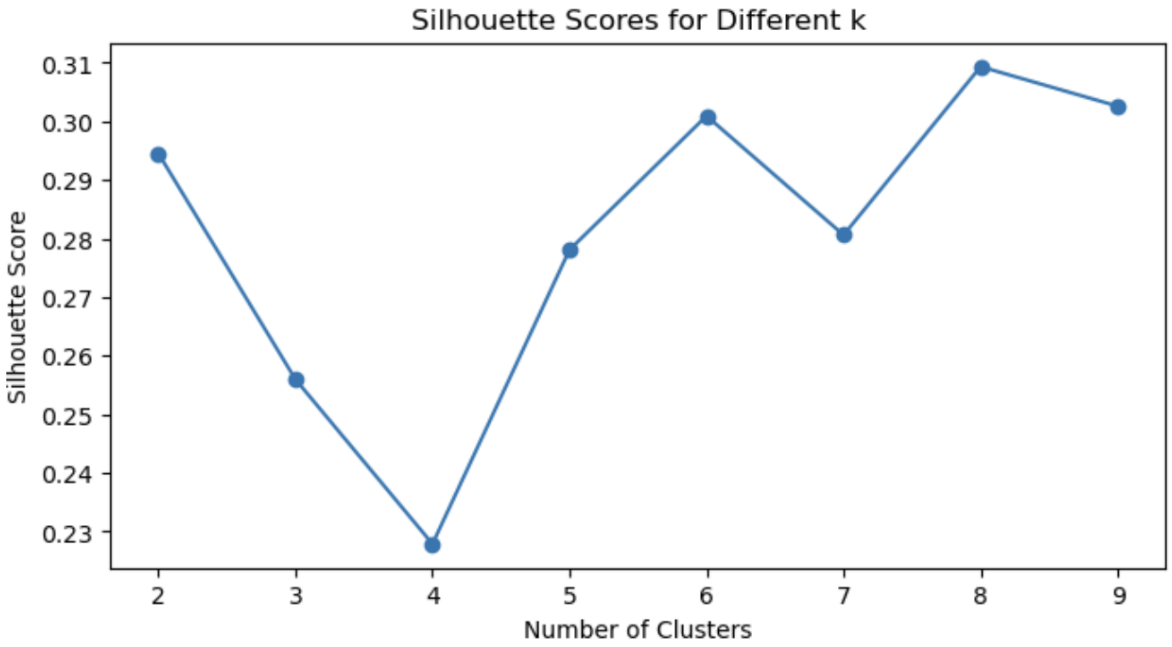
Support: The support column indicates the number of true occurrences of each class in the model. There are 6 and 75 true instances for class 1 and 2 and 593 true instances of class 3. The poor performance on class 1 and 2 might indicate to a class imbalance.

Macro Average: The precision, recall, and F1-score are averaged without taking support into account. The low macro averages for precision, recall, and F1-score = 0.39, 0.33, and 0.31, respectively—show that the model performs poorly overall in all classes.

Weighted Average: The weighted average considers each class's level of support. These are greater because of the model's outstanding performance on Class 3, which has the greatest number of instances: 0.77 for precision, 0.88 for recall, and 0.82 for F1-score.

While the accuracy of the model is high, this metric is misleading due to the imbalanced nature of the dataset. The model is effective in identifying class 3 but fails to recognize classes 1 and 2, this suggests that the decision tree maybe overfitting to the majority class and ignoring the minority classes.

K means Clustering Evaluation:



The silhouette score is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The core goes from -1 to 1 where the higher scores indicate the object is well matched to its own cluster. If most values are higher than the clustering configuration is appropriate.

Our clustering analysis identified three distinct clusters of traffic incidents based on several attributes such as location coordinates, number of casualties, road numbers and speed limits. These clusters help us understand the common characteristics of incidents occur in different areas.

```
Cluster 0 summary:
         Longitude     Latitude  Number_of_Casualties  1st_Road_Number  \
count  2083.000000  2083.000000           2083.000000      2083.000000
mean     -0.355006    51.472985              1.128661       162.477196
std       0.056272     0.018081              0.334904       162.984906
min      -0.458762    51.427842              1.000000         0.000000
25%      -0.403977    51.460920              1.000000         0.000000
50%      -0.371641    51.477276              1.000000       205.000000
75%      -0.304090    51.488056              1.000000       315.000000
max      -0.245604    51.501776              2.000000       491.000000

       Speed_limit  2nd_Road_Number  Cluster
count  2083.000000      2083.000000   2083.0
mean     31.228997        39.209313      0.0
std       3.284008       101.423563      0.0
min      30.000000         0.000000      0.0
25%      30.000000         0.000000      0.0
50%      30.000000         0.000000      0.0
75%      30.000000         0.000000      0.0
max      40.000000       316.000000      0.0
```

Cluster 0: Urban Core Incidents:

Cluster 0 represents a group of 2083 traffic incidents that predominantly occur in geographically compact area, as indicated by the mean easting and northing coordinates. These accidents share the following characteristics:

- **Location:** The mean of longitude and latitude suggest that these accidents are geographically close to each other in a specific area.
- **Casualties:** On average, there is slightly more than one casualty per accident. The maximum number of casualties in this cluster is two, which indicates that the accidents are relatively low impact in terms of casualties.
- **Road Number:** The 1st Road Number mean is 167.476 which indicates a primary road where many accidents occur. The 2nd Road Number average is low, suggesting that the secondary roads are less frequently involved.
- **Speed Limit:** The average speed is around 31 km/h, with a minimum of 30 and a maximum of 40. This lower speed limit suggests that these accidents occur in urban areas with traffic control.

```
Cluster 1 summary:
        Longitude    Latitude  Number_of_Casualties  1st_Road_Number  \
count  842.000000  842.000000            842.000000       842.000000
mean    -0.357226   51.472807              1.155582       164.219715
std      0.055106    0.017728              0.362674       163.230374
min     -0.457489   51.428624              1.000000         0.000000
25%     -0.405400   51.462068              1.000000         0.000000
50%     -0.374948   51.476793              1.000000       205.000000
75%     -0.309214   51.487547              1.000000       315.000000
max     -0.247191   51.498729              2.000000       491.000000

       Speed_limit  2nd_Road_Number  Cluster
count   842.000000       842.000000    842.0
mean     31.413302        45.669834      1.0
std       3.485689       107.637246      0.0
min      30.000000         0.000000      1.0
25%      30.000000         0.000000      1.0
50%      30.000000         0.000000      1.0
75%      30.000000         3.000000      1.0
max      40.000000       316.000000      1.0
```

Cluster 1: Peripherals Urban Incidents:

Cluster one has a count of 842 meaning the cluster has 842 accidents. It appears to be representing areas on the periphery of the urban core as the statistics is similar to Cluster 0 but with slight variations.

- **Location:** The mean suggests the location coordinates are marginally different from Cluster 0, suggesting proximity to the core urban incidents but perhaps slightly more spread out as indicated by standard deviation.
- **Casualties and Speed Limit:** Similar to Cluster 0 as there is an average of just over 1 casualty per accident and low average speed limit suggests urban area with significant traffic control.
- **Road Number:** The road numbers also are not significantly different from cluster 0, which suggests these accidents occurred in the same type of road.

```
Cluster 2 summary:
        Longitude    Latitude  Number_of_Casualties  1st_Road_Number  \
count  442.000000  442.000000            442.000000       442.000000
mean    -0.356849   51.471866              1.160633       167.194570
std      0.055190    0.017652              0.367609       160.879649
min     -0.457486   51.429860              1.000000         0.000000
25%     -0.403820   51.459606              1.000000         0.000000
50%     -0.374147   51.475894              1.000000       244.000000
75%     -0.307068   51.487009              1.000000       315.000000
max     -0.246324   51.498476              2.000000       454.000000

       Speed_limit  2nd_Road_Number  Cluster
count   442.000000       442.000000    442.0
mean     31.742081        29.142534      2.0
std       3.797182        88.220735      0.0
min      30.000000         0.000000      2.0
25%      30.000000         0.000000      2.0
50%      30.000000         0.000000      2.0
75%      30.000000         0.000000      2.0
max      40.000000       316.000000      2.0
```

Cluster 2: Adjacent Urban Incidents:

Cluster 1, with 442 incidents with characteristics indicating that these incidents occur in areas adjacent to the clusters mentioned above along with some distinct features.

- **Location & Spread:** While the location is the quite close to the other clusters, the spread is wider, indicating a more diverse set of incident locations that might extend slightly beyond the core and peripheral urban areas.
- **Speed Limit:** The mean speed limit is higher than Cluster 0 and 1, which suggests incidents occurring on roads that allow slightly higher speeds.
- **Casualties:** The number of casualties is slightly higher than in the other two clusters but still close to one.

Insights:

- **Geographic Concentration:** All three clusters are geographically close and indicating a specific urban area where these accidents occurred.
- **Casualty Consistency:** The number of casualties is consistent across clusters, with most accidents resulting in one to two casualties, indicating fatal accidents are rare.
- **Speed Limit:** The speed limit across all clusters is low, with little variation, which indicates the accident mostly occur in busy roads with traffic.
- **Road Numbers:** There seems to be a commonality in the accidents occurring on primary roads while secondary roads are less involved.

Possible Actions:

- **Safety Improvements:** Given that most accidents are occurring on primary roads within a specific urban area, local authorities could investigate these roads for potential safety improvements.
- **Speed Limit Enforcement:** Since the speed limits are already low, ensuring that these limits are strictly enforced could help reduce the number of accidents.

- **Urban Planning:** The data might suggest that need for better urban planning, with a focus on accident hotspots identified by the clusters.

Conclusion:

The multifaceted data analysis project has provided a comprehensive overview of factors contributing to road accident severity. Utilizing a Decision Tree Classifier illuminated the impact of weather conditions and the number of vehicles on the severity of accidents, revealing a nuanced relationship that extends beyond mere numerical correlations. The accuracy and classification report from the model underscored its predictive validity, albeit with an emphasis on further validation to ensure robustness against imbalanced data.

Parallel to this, the K-Means clustering unearthed spatial trends, identifying hotspots within the Hounslow borough that are particularly prone to higher accident frequencies. This geospatial analysis is crucial for prioritizing areas for traffic safety improvements and can be instrumental for local authorities in implementing preventive measures.

Furthermore, the hotspot maps gave us great insights on which areas in the boroughs have high number of accidents and highways like A4 where high numbers of accidents were recorded.

In synthesis, the project's findings from various modeling approaches form a constellation of insights. They together advocate for a data-driven strategy in addressing road safety, where predictive models can forecast accident severity and clustering analyses can spotlight regions for intervention. The resulting narrative is clear: leveraging such analytics can significantly contribute to the proactive management of road safety and the optimization of emergency services.

Bibliography:

1. R Anusha Shetty and Indiramma (2021). Clustering Analysis of Traffic Accident Dataset using Canopy K Means. *2021 IEEE Mysore Sub Section International Conference (MysuruCon)*. doi:https://doi.org/10.1109/mysurucon52639.2021.9641638.

2. Witten, I.H. and Frank, E. (2002). Data mining. *ACM SIGMOD Record*, 31(1), p.76. doi:https://doi.org/10.1145/507338.507355.

3. Han, J., Kamber, M. and Pei, J. (2011). *C o nc epts a nd T ec hniques -C ha pter 2*. [online] Available at: https://liacs.leidenuniv.nl/~bakkerem2/dbdm2012/03_dbdm2012_Data.pdf.